Reminders  Collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.

Exercises  These should not be handed in, but the material they cover may appear on exams.

- CLRS (textbook) problem 1-1, exercises 2.1-3 and 2.2-2.
- The “exchange” operation swaps two adjacent elements of an array. Rewrite the pseudocode for insertion sort using “exchange” as the only operation that modifies the array (that is, you can read array elements, but not write directly into the array). How does the running time of the modified algorithm compare to the running time of the original?
- Rewrite insertion sort as a recursive algorithm that first makes a recursive call to sort an array of size \( n - 1 \) and then inserts the remaining element in the correct location. How does the running time of the modified algorithm compare to the running time of the original?

Problems to be handed in

1. CLRS problem 2-2 (Correctness of bubblesort). Add the following, final part:

   e. For each statement below, either prove it (by giving an example) or explain why it cannot be true.

      i. There exists a permutation of integers 1 to \( n \) on which bubblesort takes time \( \Theta(n^2) \), but insertion sort takes time only \( \Theta(n) \).
      ii. There exists a permutation of integers 1 to \( n \) on which insertion sort takes time \( \Theta(n^2) \), but bubblesort takes time only \( \Theta(n) \).

2. (Anagrams). Consider the following problem: Given two words, determine whether they are anagrams, that is if one word can be obtained by permuting the letters of the other. (For example, “stop” and “tops” are anagrams; “mutt” and “tumm” are not.) The input is given as a pair of arrays of English letters, and its length, \( n \), is the sum of the lengths of the words.

   For each your algorithms, explain the idea in one or two sentences; give pseudocode for the algorithm; prove that the algorithm is correct; and analyze it’s running time. Use big-\( \Theta \) notation to express running times.

   (a) Using sorting as a subroutine, give an algorithm for the anagrams problem. When analyzing running time, you may assume that the sorting subroutine takes time \( \Theta(n \log n) \) on inputs of length \( n \).
(b) Give a $\Theta(n)$-time algorithm for the anagrams problem (hint: think about counting).

[N.B: There will also be a programming problem to do this week that will be due separately.]